



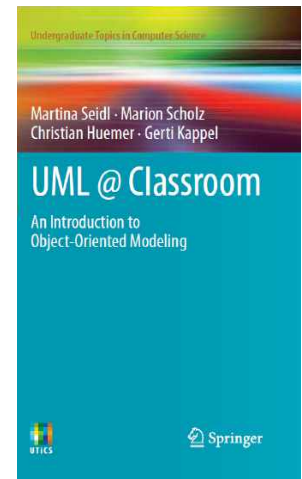
# Object-Oriented Modeling

## State Machine Diagram

Slide untuk melengkapi buku UML@Classroom  
Versi 1.0

Diterjemahkan dari

slide milik Business Informatics Group, Vienna University of Technology



**Program Studi Teknik Informatika**

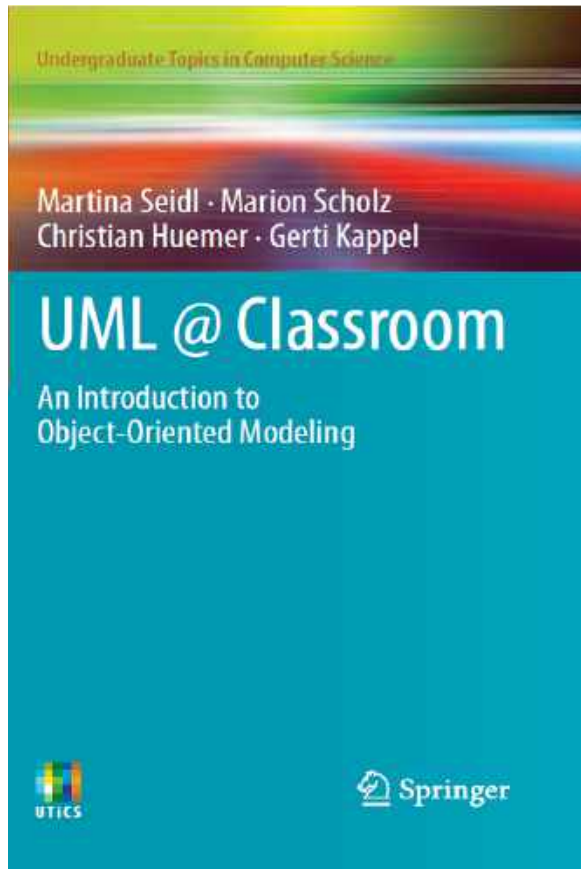
Jurusan Teknik Informatika  
Politeknik Negeri Batam

Jalan Ahmad Yani, Batam Center, Batam 29461  
[www.polibatam.ac.id](http://www.polibatam.ac.id)

# Pustaka

---

- Materi kuliah diambil dari buku berikut:



## **UML @ Classroom: An Introduction to Object-Oriented Modeling**

Martina Seidl, Marion Scholz, Christian Huemer and Gerti Kappel

Springer Publishing, 2015

ISBN 3319127411

- Use Case Diagram
- Structure Modeling
- **State Machine Diagram**
- Sequence Diagram
- Activity Diagram

# Materi

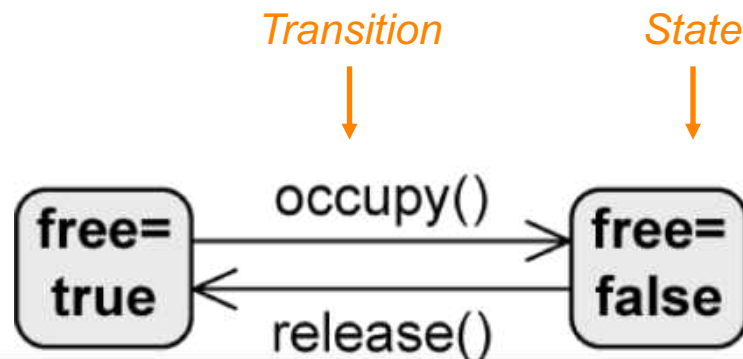
---

- Pengenalan
- *State*
- *Transition*
- Tipe *event*
- Tipe *state*
- *Entry* dan *exit points*

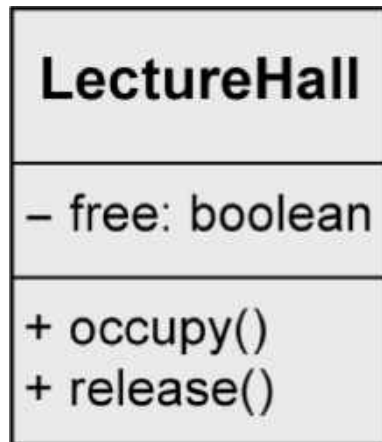
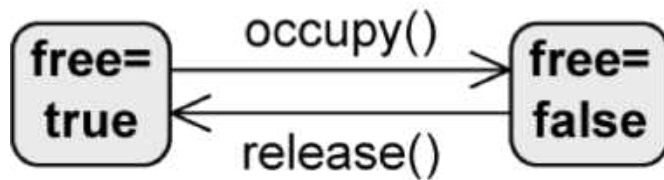
**Catatan:** materi kuliah ini menggunakan istilah-istilah dalam bahasa aslinya yaitu Bahasa Inggris untuk menghindari kesalahan arti

# Pengenalan

- Setiap *object* dapat mengalami sejumlah state dalam masa hidupnya
- State machine diagram digunakan sebagai berikut:
  - Untuk memodelkan berbagai state yang mungkin dialami oleh sebuah sistem atau *object*
  - Untuk menunjukkan bagaimana perubahan state/*state transition* terjadi karena adanya event
  - Untuk menunjukkan perilaku sistem atau object dalam setiap state
- Contoh: deskripsi perilaku sebuah *lecture hall*



## Contoh: *Lecture Hall* (Ruang Kuliah) dengan Detail

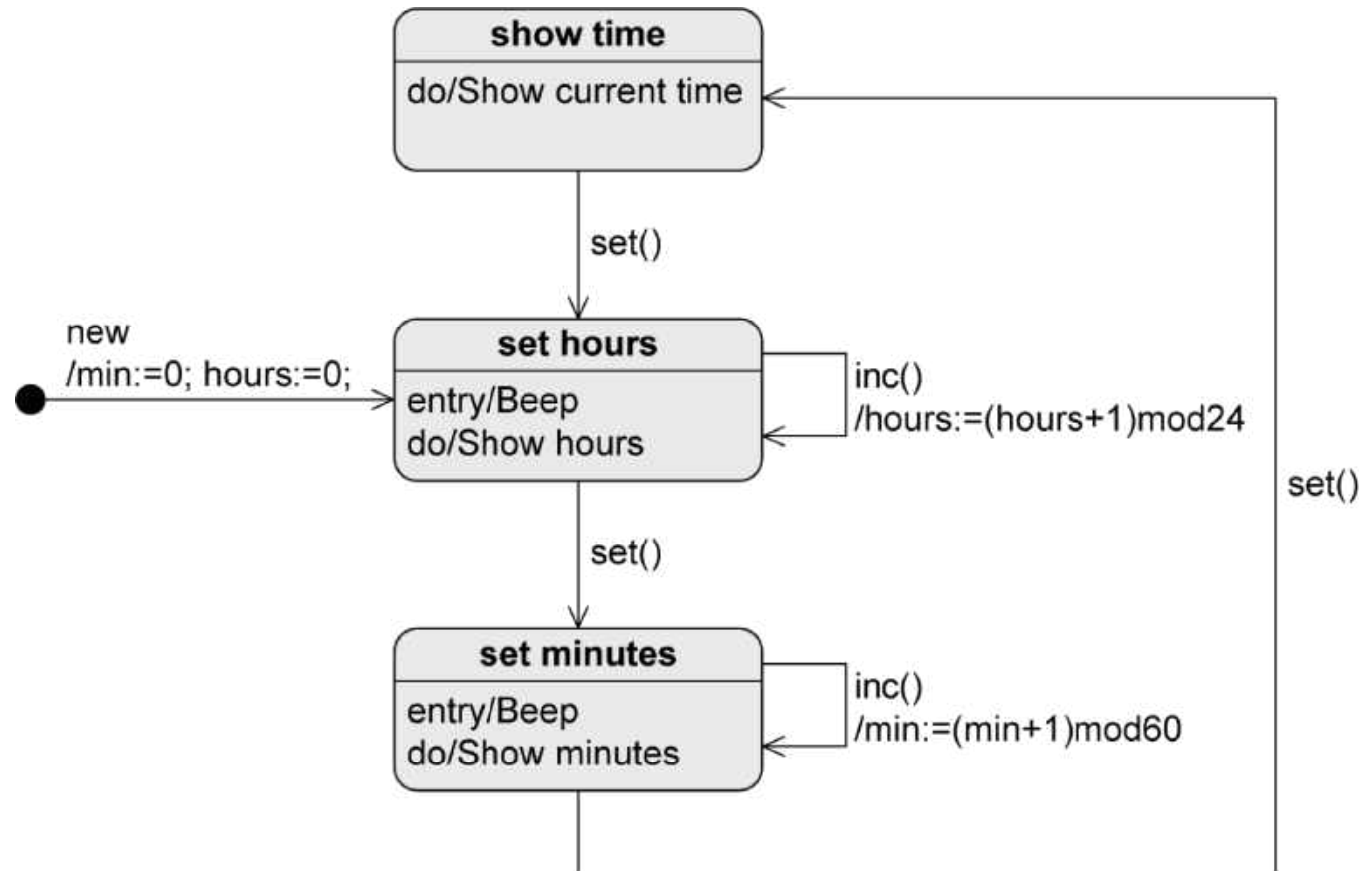


```
class LectureHall {  
    private boolean free;  
  
    public void occupy() {  
        free=false;  
    }  
    public void release() {  
        free=true;  
    }  
}
```

## Contoh: *Digital Clock* (Jam Digital)

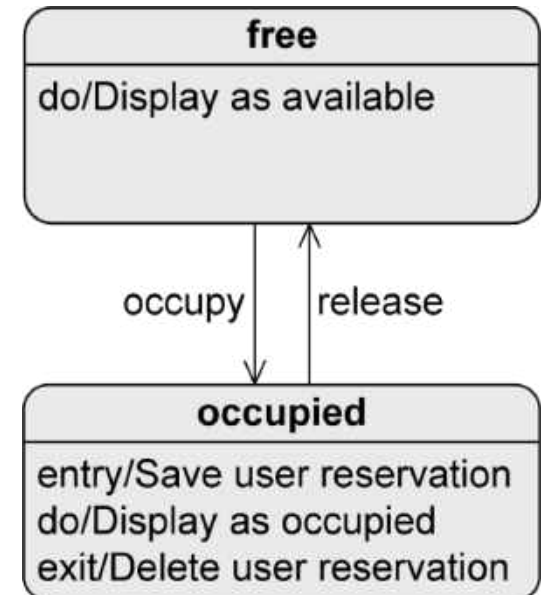
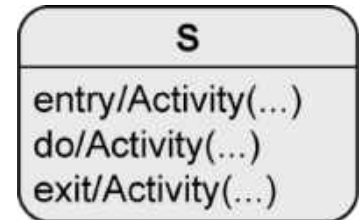
00 : 00 ● set  
● inc

DigitalClock
- min: int - hours: int
+ set(): void + inc(): void

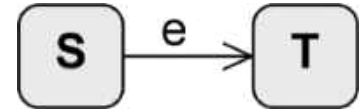


# State

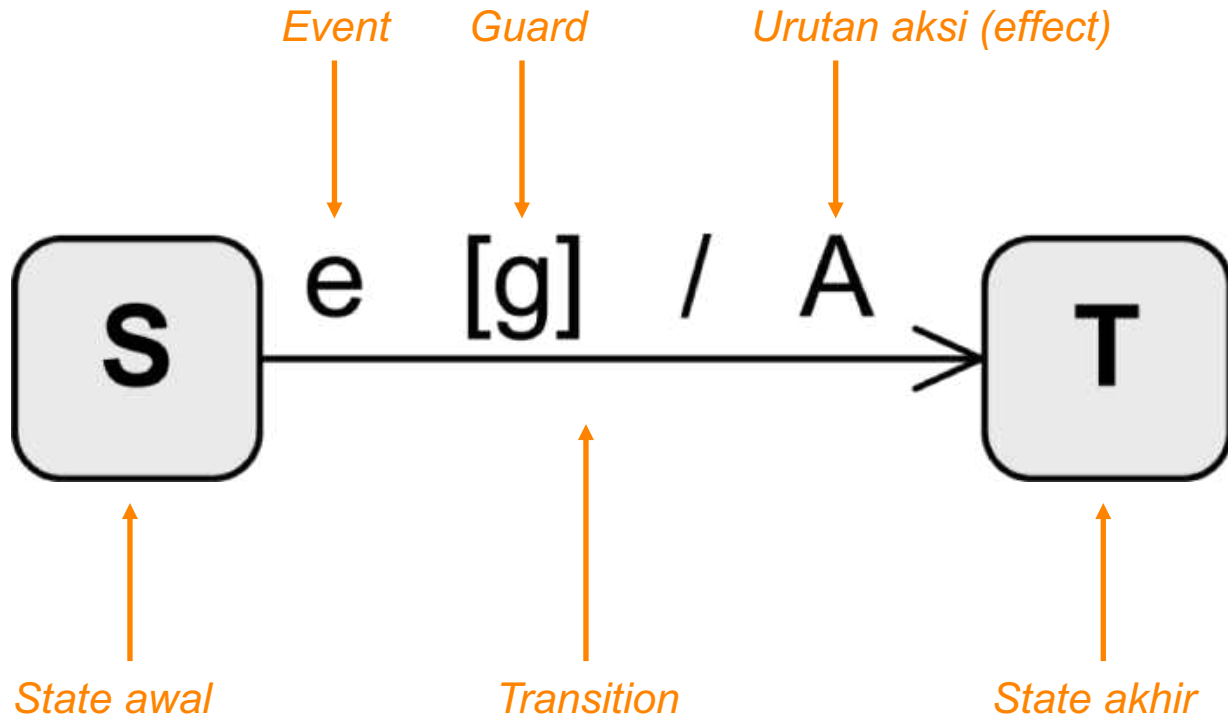
- States = simpul pada state machine
- Jika sebuah state menjadi aktif
  - Object berada pada state tersebut
  - Semua aktivitas internal yang dispesifikasikan pada state ini dapat dijalankan
    - Sebuah aktivitas dapat terdiri dari beberapa langkah aksi
- entry / Activity(...)
  - Dijalankan saat object memasuki state tersebut
- exit / Activity(...)
  - Dijalankan saat object keluar dari state tersebut
- do / Activity(...)
  - Dijalankan selama object masih berada dalam state tersebut



# Transition

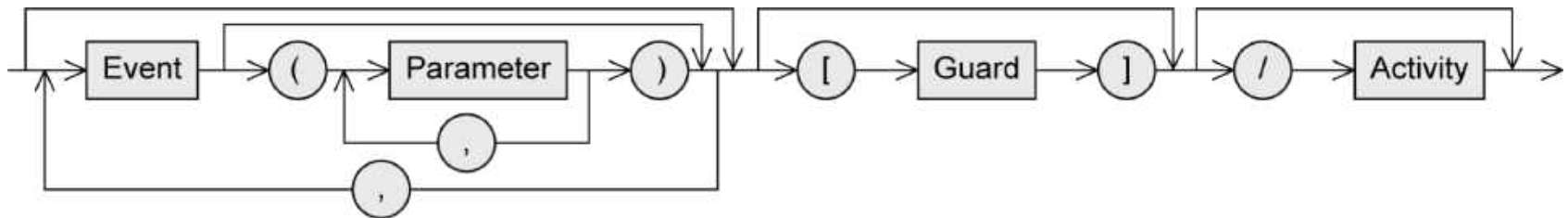


- Perubahan dari satu state ke state yang lain





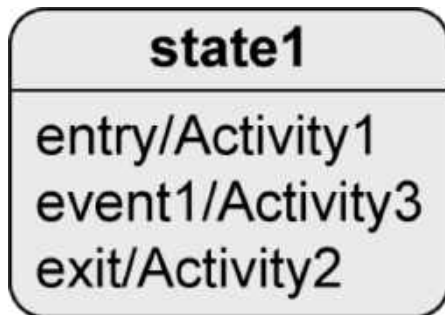
# Transition – Syntax



- **Event** (trigger)
  - Stimulus eksogen (berasal dari luar sistem)
  - Dapat memicu sebuah *state transition*
- **Guard** (condition)
  - Ekspresi Boolean
  - Jika event terjadi, guard akan diperiksa
  - Jika guard bernilai true
    - Semua aktivitas dalam state sekarang dihentikan
    - Semua aktivitas *exit* yang relevan dijalankan
    - *Transition*/perubahan state terjadi
  - Jika guard bernilai false
    - Tidak ada state transition yang terjadi, event “dibuang”
- **Activity** (effect)
  - Urutan aksi yang dijalankan selama *state transition*

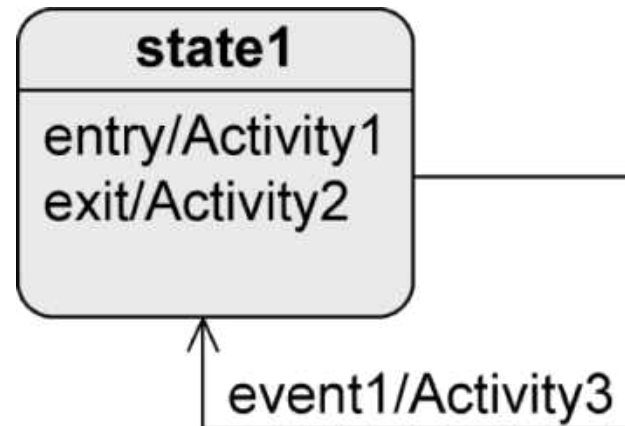
## Transition – Tipe (1/2)

### Internal transition



- Jika **event1** terjadi
  - Object tetap di **state1**
  - **Activity3** dijalankan

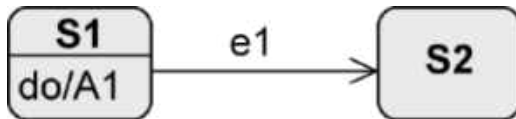
### External transition



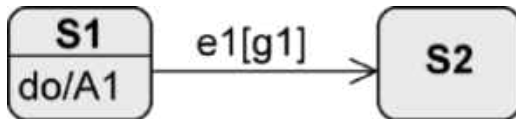
- Jika **event1** terjadi
  - Object keluar dari **state1** dan **Activity2** dijalankan
  - **Activity3** dijalankan
  - Object memasuki **state1** dan **Activity1** dijalankan

## Transition – Tipe (2/2)

- Kapan transition berikut terjadi?



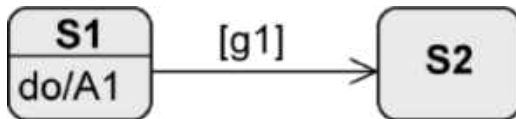
Jika **e1** terjadi, **A1** dihentikan dan object berpindah ke **S2**



Jika **e1** terjadi dan **g1** bernilai true, **A1** dihentikan dan object berpindah ke **S2**



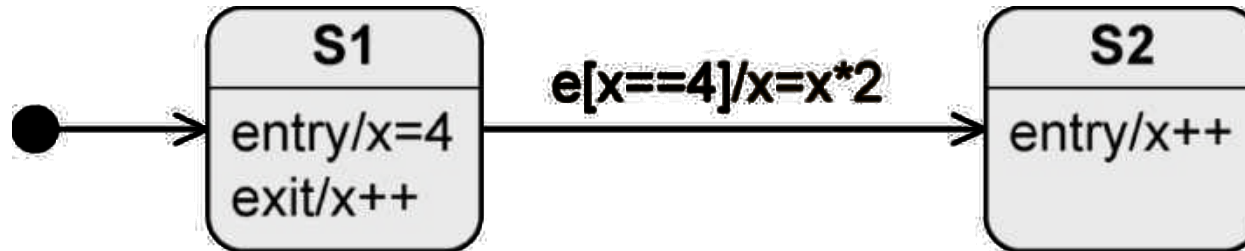
Segera setelah **A1** selesai dijalankan, *completion event* yang menyebabkan transition ke **S2** dimulai



Segera setelah **A1** selesai dijalankan, *completion event* dimulai; jika **g1** bernilai true, transition terjadi; jika false, transition tidak akan pernah terjadi

## Transition – Urutan Aktivitas Dijalankan

- Asumsi **S1** aktif ... berapa nilai **x** setelah **e** terjadi?



**S1** menjadi aktif, **x** diisi nilai 4

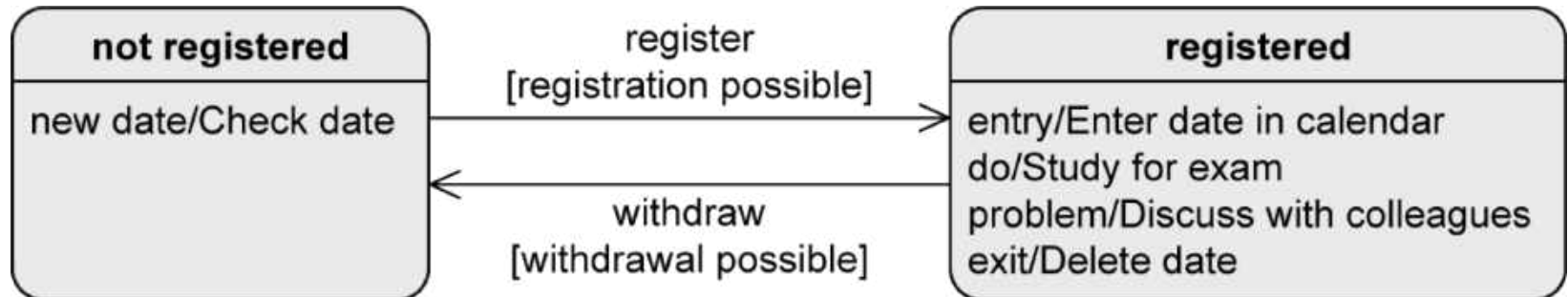
**e** terjadi, guard diperiksa dan hasilnya bernilai true

**S1** ditinggalkan, **x** diisi nilai 5

Transition terjadi, **x** diisi nilai 10

Masuk ke **S2**, **x** diisi nilai 11

## Contoh: Status *Registration* untuk sebuah *Exam*



## Event – Tipe (1/2)

---

- **Signal event**

Saat menerima sebuah signal

- Misal **rightmousedown**, **sendSMS(message)**

- **Call event**

Operation call

- E.g., **occupy(user, lectureHall)**, **register(exam)**

- **Time event**

State transition berdasarkan waktu

- Relative: berdasarkan waktu terjadinya sebuah event
  - Misal **after(5 seconds)**
- Absolute
  - Misal **when(time==16:00)**, **when(date==20150101)**

## Event – Tipe (2/2)

- **Any receive event**

Terjadi saat sebuah event terjadi tapi tidak menyebabkan terjadinya transition dari state yang sedang aktif

- Keyword **all**

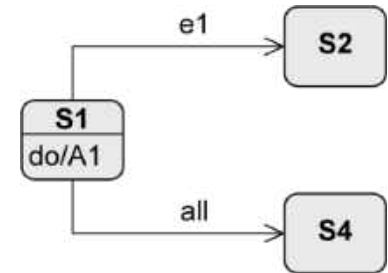
- **Completion event**

Terjadi secara otomatis pada saat semua yang harus dilakukan di sebuah state sudah selesai dilakukan

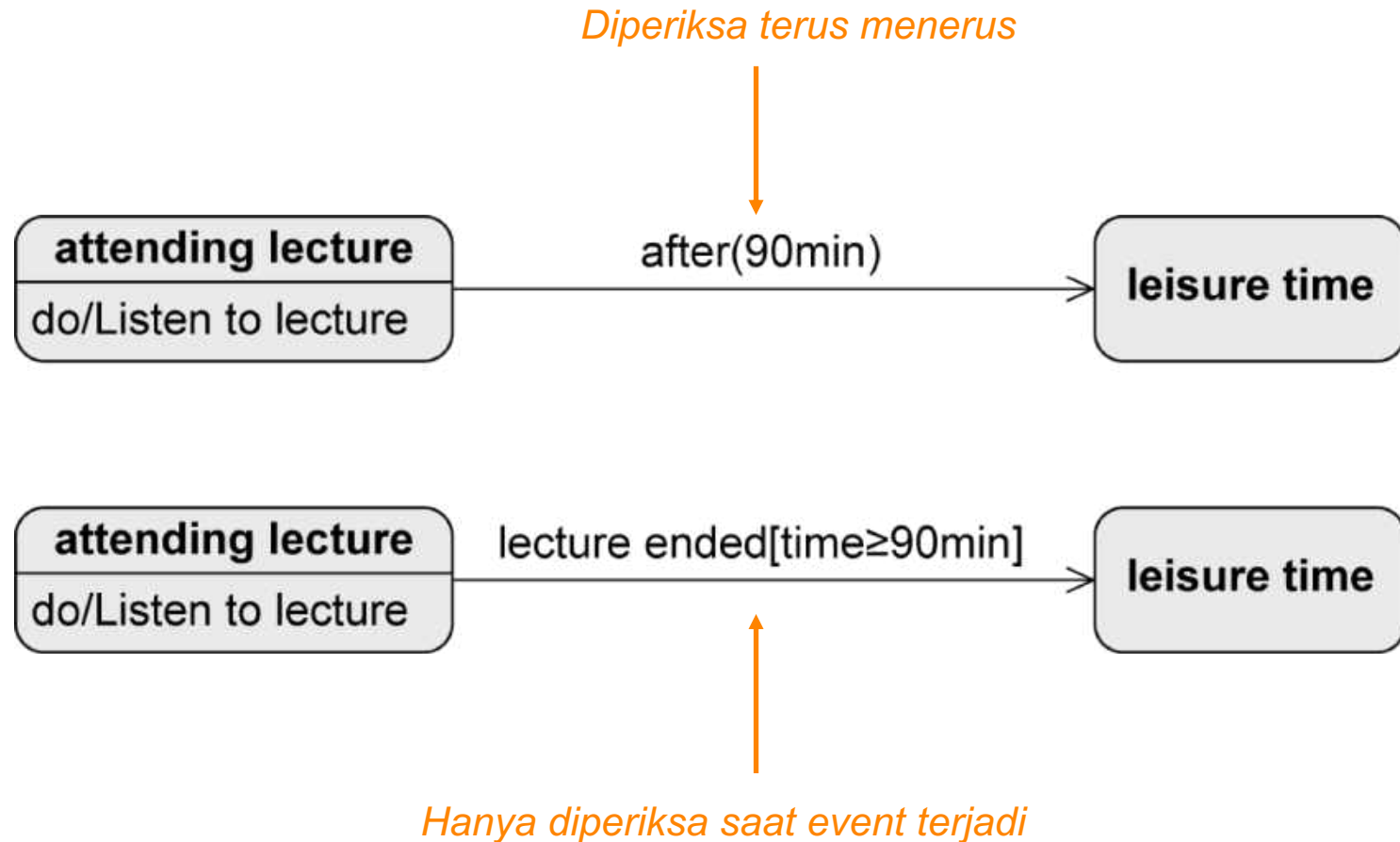
- **Change event**

Secara terus menerus memeriksa apakah sebuah *condition* berubah nilai menjadi true

- Misal **when(x > y), after(90min)**



## Change Event vs. Guard



*Pertanyaan: Bagaimana jika lecture kurang dari 90min?*



# Initial State

---

- “Start” of a state machine diagram
- Pseudostate
  - Transient, i.e., system cannot remain in that state
  - Rather a control structure than a real state
- No incoming edges
- If  $>1$  outgoing edges
  - Guards must be mutually exclusive and cover all possible cases to ensure that exactly one target state is reached
- If initial state becomes active, the object immediately switches to the next state
  - No events allowed on the outgoing edges (exception: **new()**)

# Final State and Terminate Node

---

## Final State



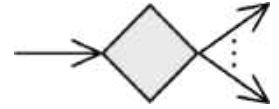
- Real state
- Marks the end of the sequence of states
- Object can remain in a final state forever

## Terminate Node

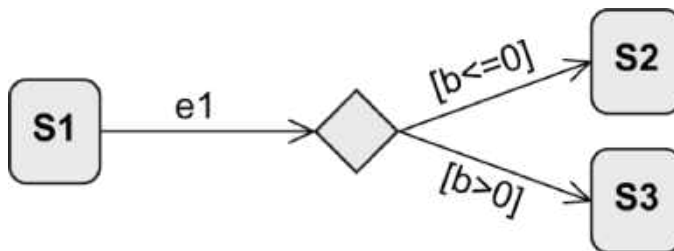


- Pseudostate
- Terminates the state machine
- The modeled object ceases to exist (= is deleted)

# Decision Node

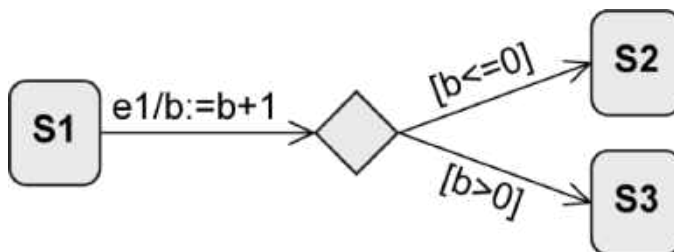
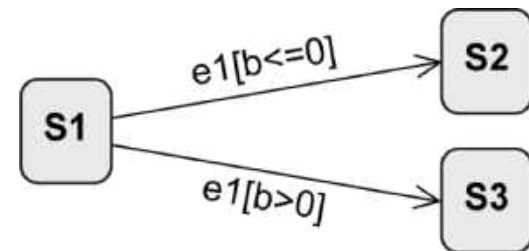


- *Pseudostate*/bukan state yang sebenarnya
- Digunakan untuk memodelkan *alternative transition*



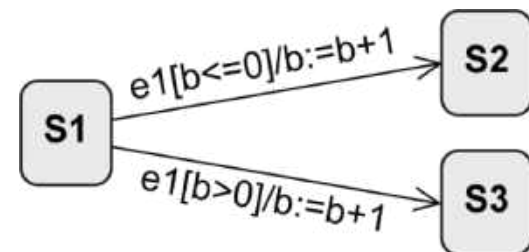
*equivalent?*

=

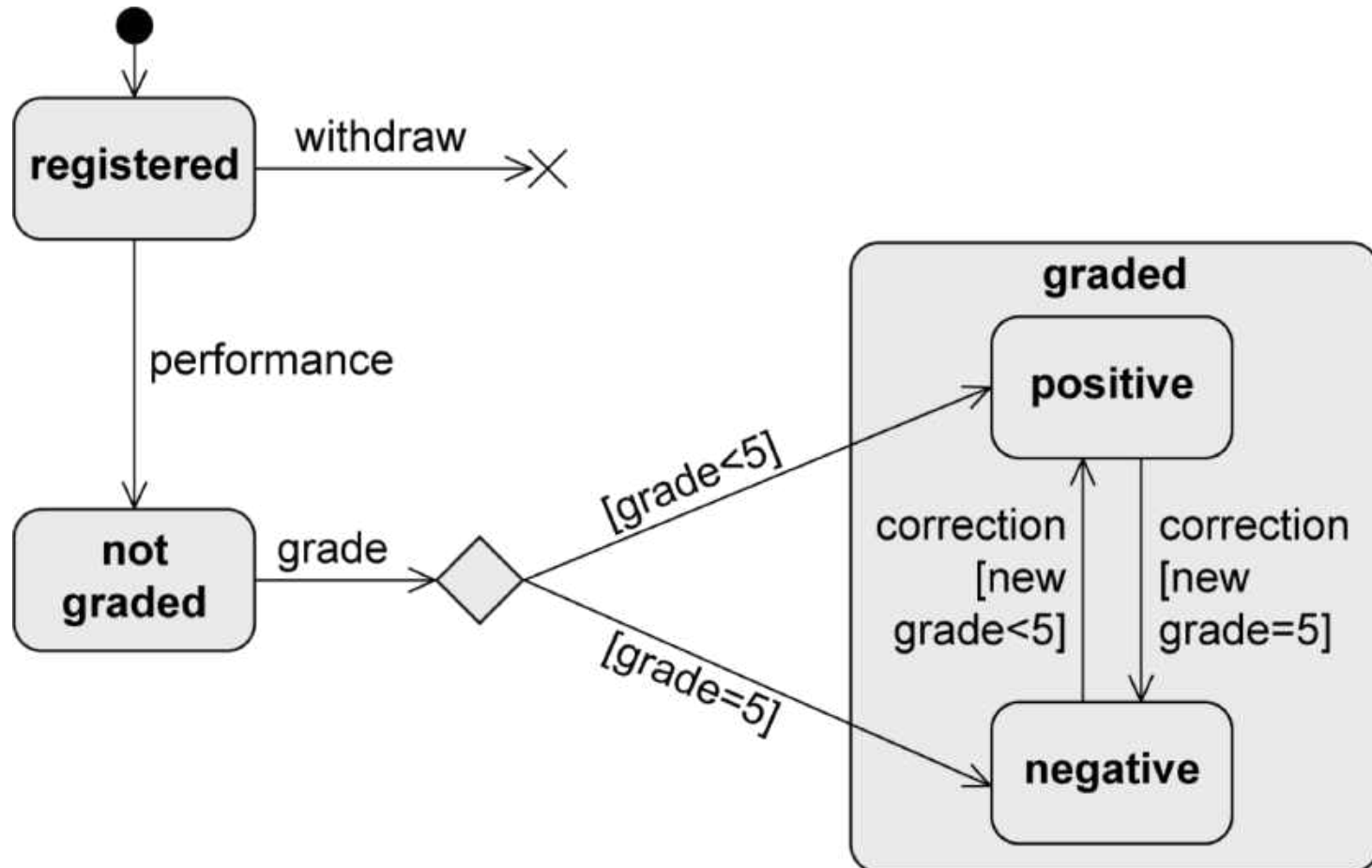


*equivalent?*

≠



## Contoh: Decision Node

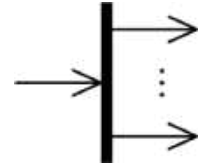


# Parallelization dan Synchronization Node

---

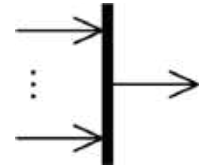
## Parallelization node

- *Pseudostate*
- Membagi control flow menjadi beberapa concurrent flows
- 1 edge masuk
- >1 edges keluar



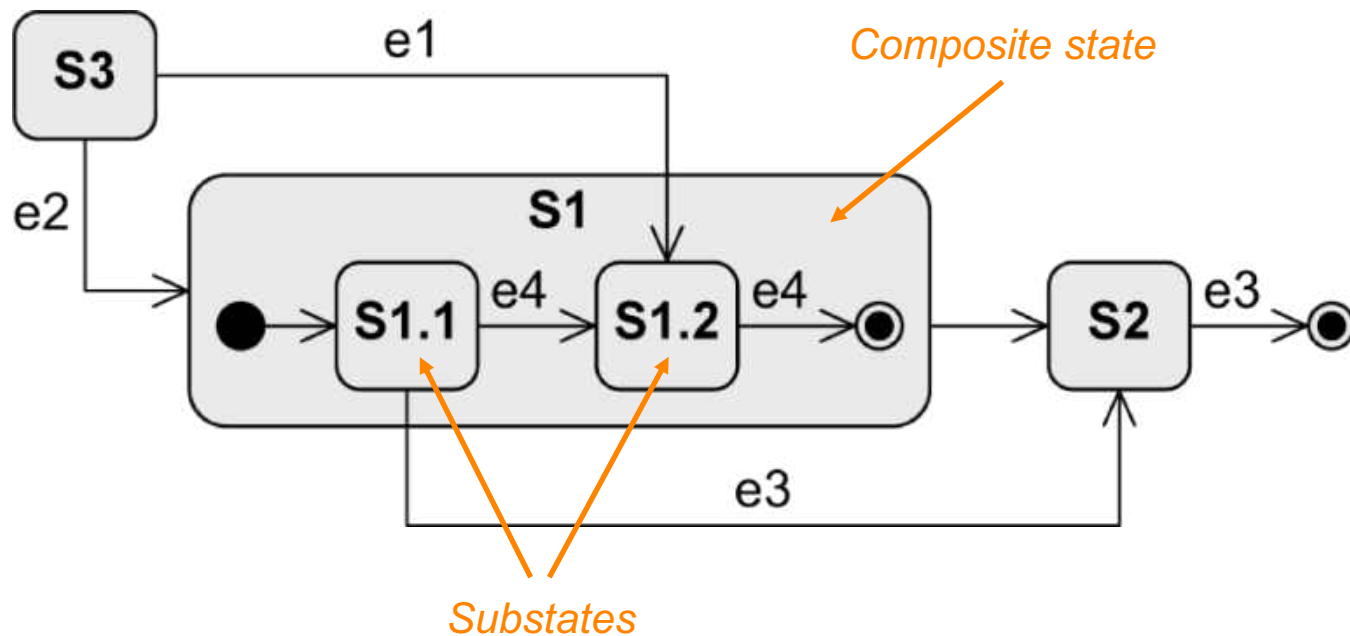
## Synchronization node

- *Pseudostate*
- Menggabungkan beberapa concurrent flows
- >1 edges masuk
- 1 edge keluar



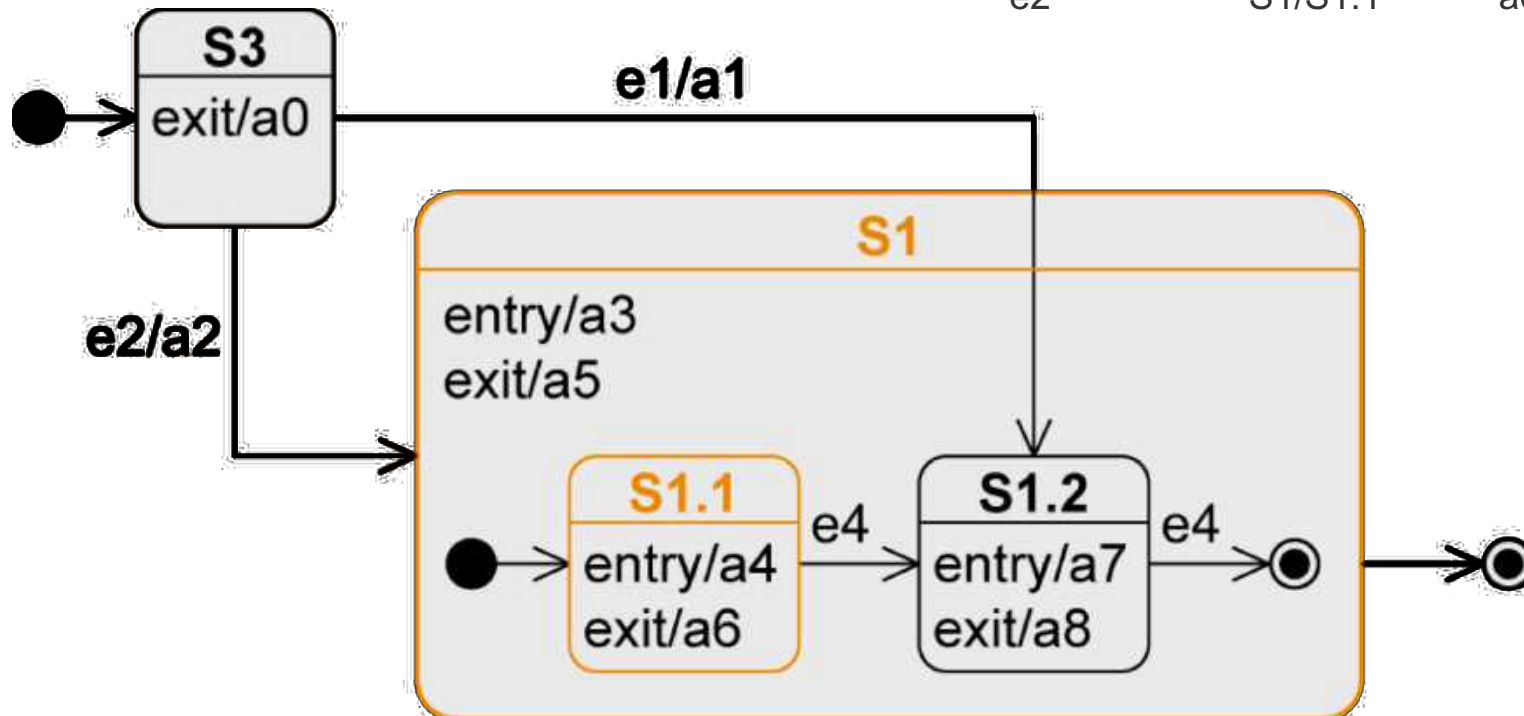
# Composite State

- Sinonim: complex state, nested state
- Berisi state yang lain – “substates”
  - Hanya satu substates yang bisa aktif pada suatu saat
- Kedalaman *nesting*/bersarang dari substates tidak ditentukan batasnya



## Memasuki Composite State (1/2)

- Transition sampai ke batas
  - Node awal dari composite state diaktifkan



Event

State

Executed  
Activities

„Beginning“

S3

e2

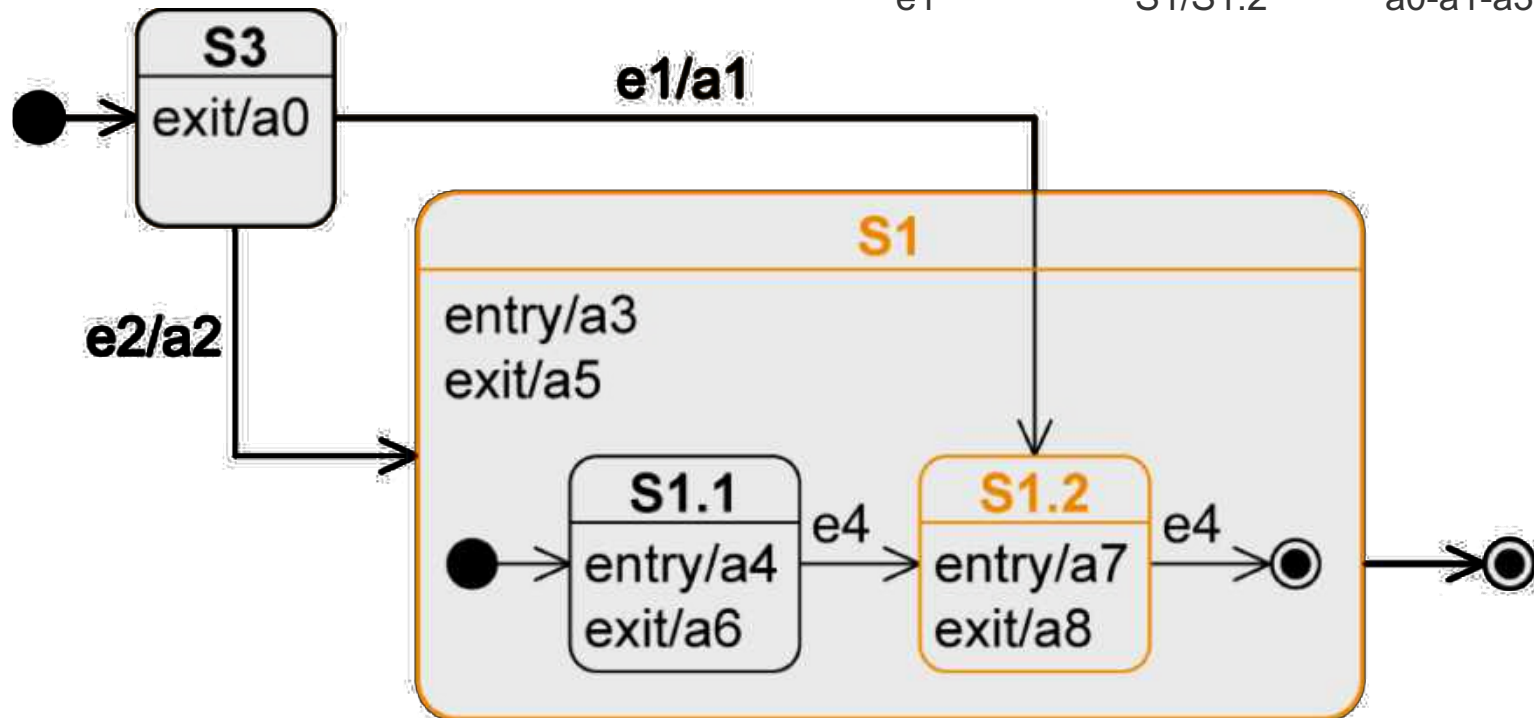
S1/S1.1

a0-a2-a3-a4

## Memasuki Composite State (2/2)

- Transition sampai ke substate
  - Substate diaktifkan

Event	State	Executed Activities
„Beginning“	S3	
e1	S1/S1.2	a0-a1-a3-a7

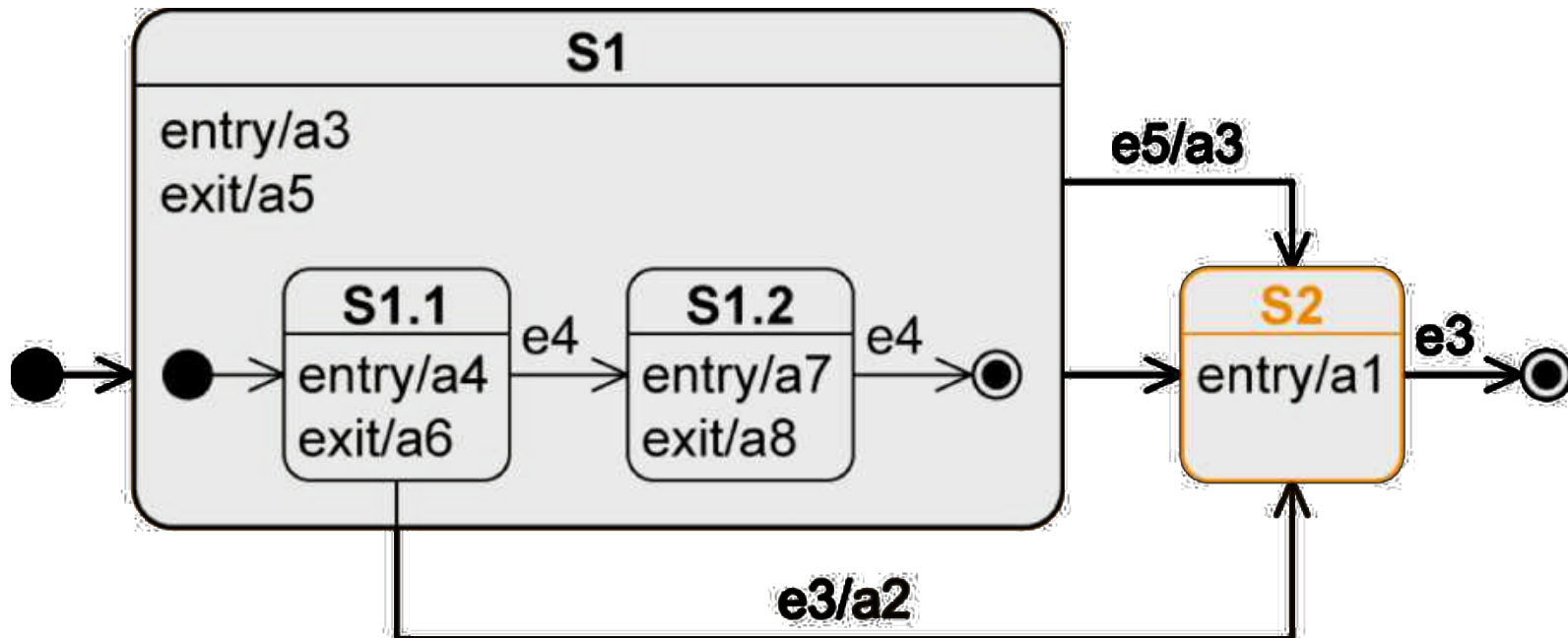




## Keluar dari Composite State (1/3)

- Transition dari substate

Event	State	Executed Activities
„Beginning“	S1/S1.1	a3-a4
e3	S2	a6-a5-a2-a1

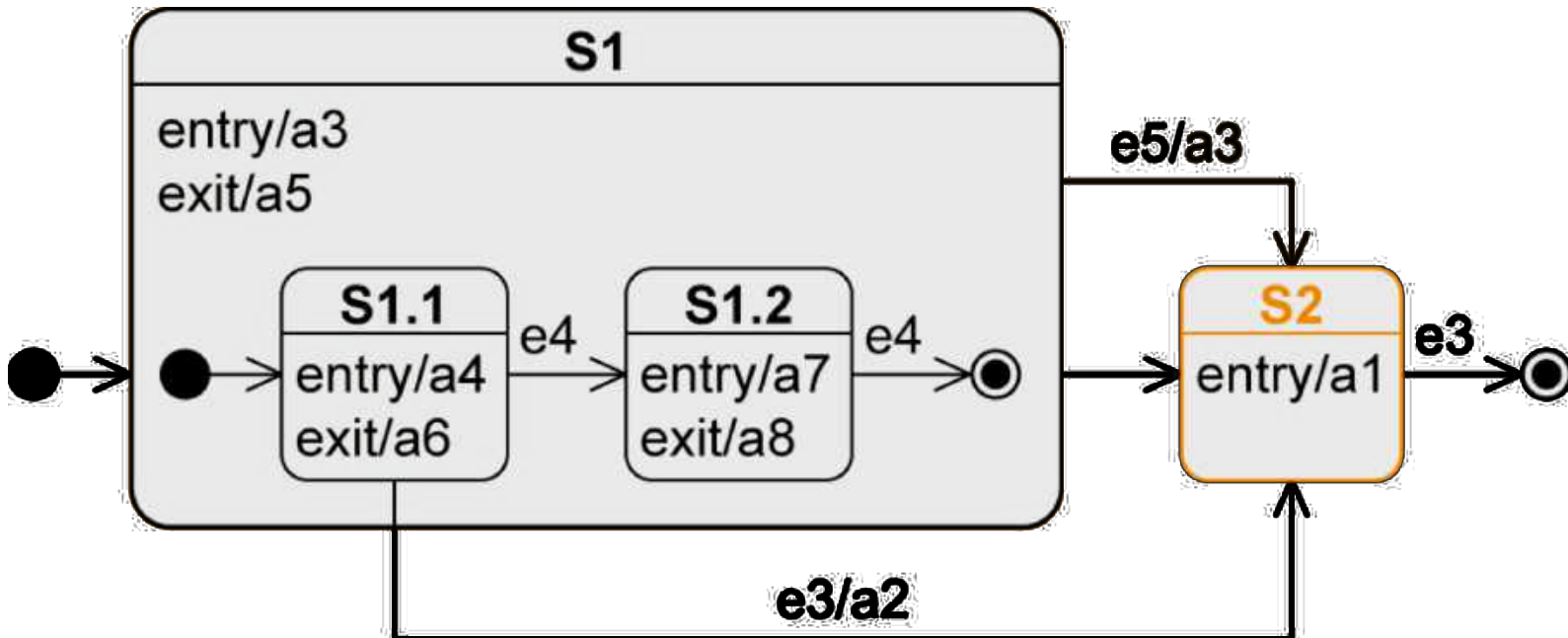


## Keluar dari Composite State (2/3)

- Transition dari composite state

*Tidak peduli substate S1 mana yang aktif, segera setelah e5 terjadi, sistem berubah ke S2*

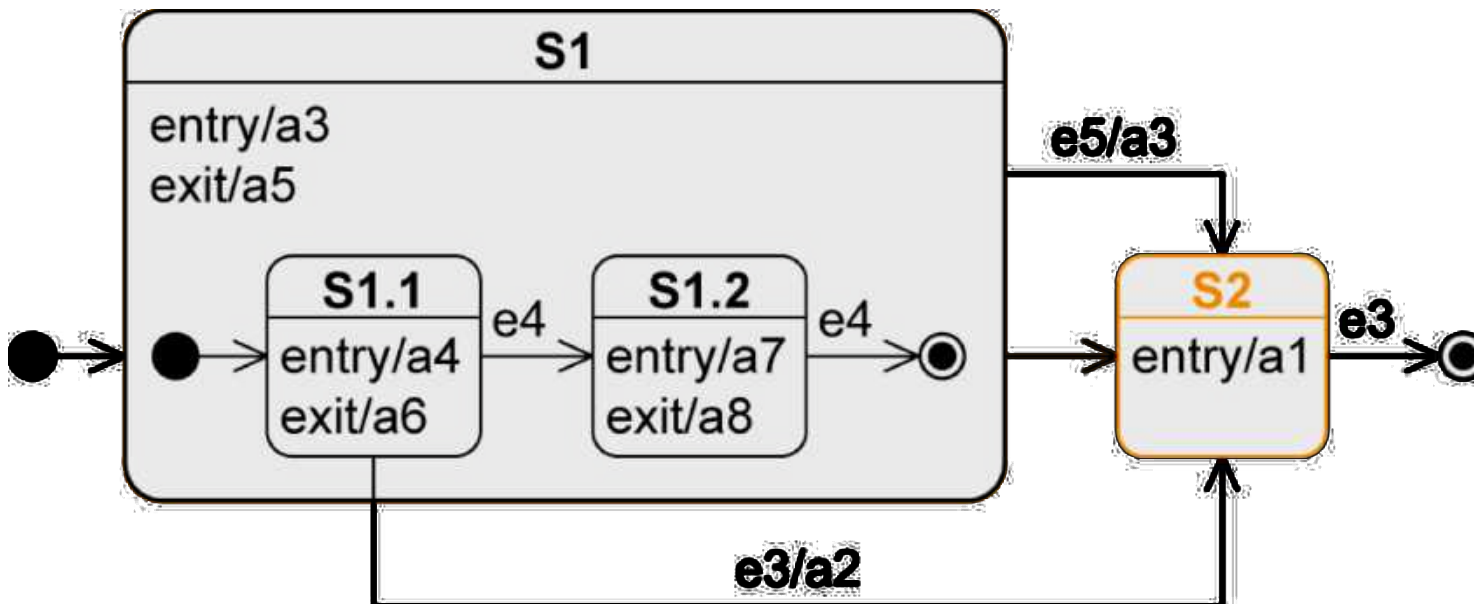
Event	State	Executed Activities
„Beginning“	S1/S1.1	a3-a4
e5	S2	a6-a5-a3-a1



## Keluar dari Composite State (3/3)

- Completion transition dari composite state

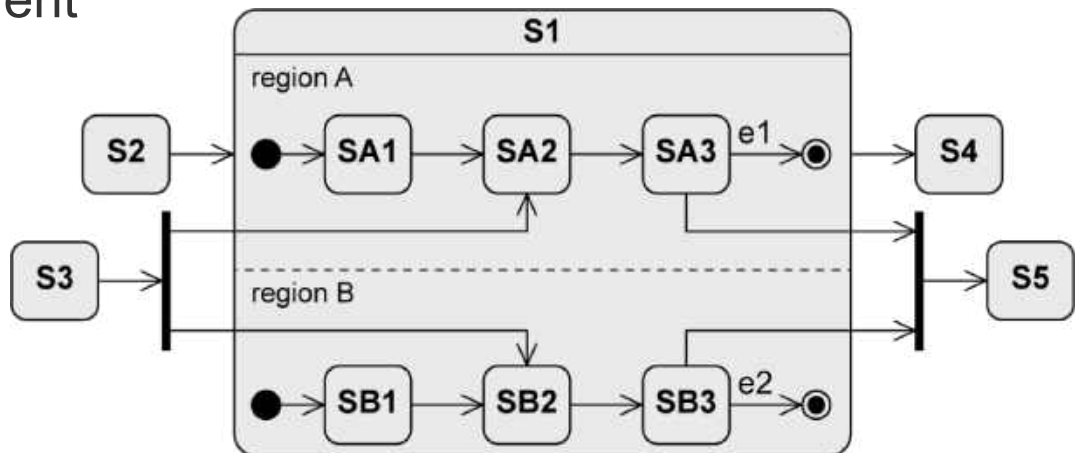
Event	State	Executed Activities
„Beginning“	S1/S1.1	a3-a4
e4	S1/S1.2	a6-a7
e4	S2	a8-a5-a1



# Orthogonal State

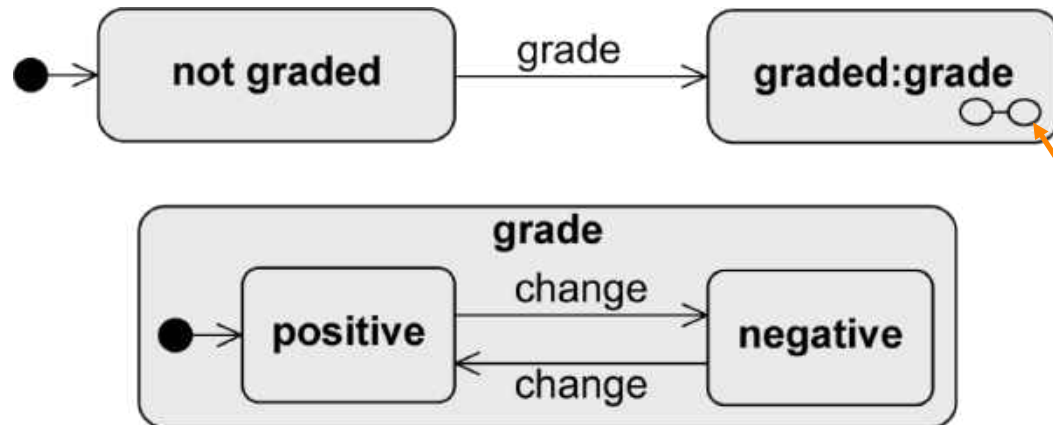
- Composite state dibagi menjadi dua atau lebih bagian/*region* yang dipisahkan dengan garis putus-putus
- Sebuah state dari setiap *region* selalu aktif pada suatu saat, yaitu concurrent substates
- **Entry:** transition ke batas dari orthogonal state mengaktifkan state awal dari semua region
- **Exit:** state akhir harus dicapai dari semua region untuk memicu terjadinya completion event

*Menggunakan parallelization dan synchronization node untuk memasuki berbagai substates*



## Submachine State (SMS)

- Untuk menggunakan kembali bagian-bagian state machine diagram dalam state machine diagram yang lain
- Notasi: **state:submachineState**
- Segera setelah sebuah submachine state diaktifkan, perilaku dari submachine dijalankan
  - Seperti memanggil subroutine dalam bahasa pemrograman



*Refinement symbol  
(optional)*

# History State

---

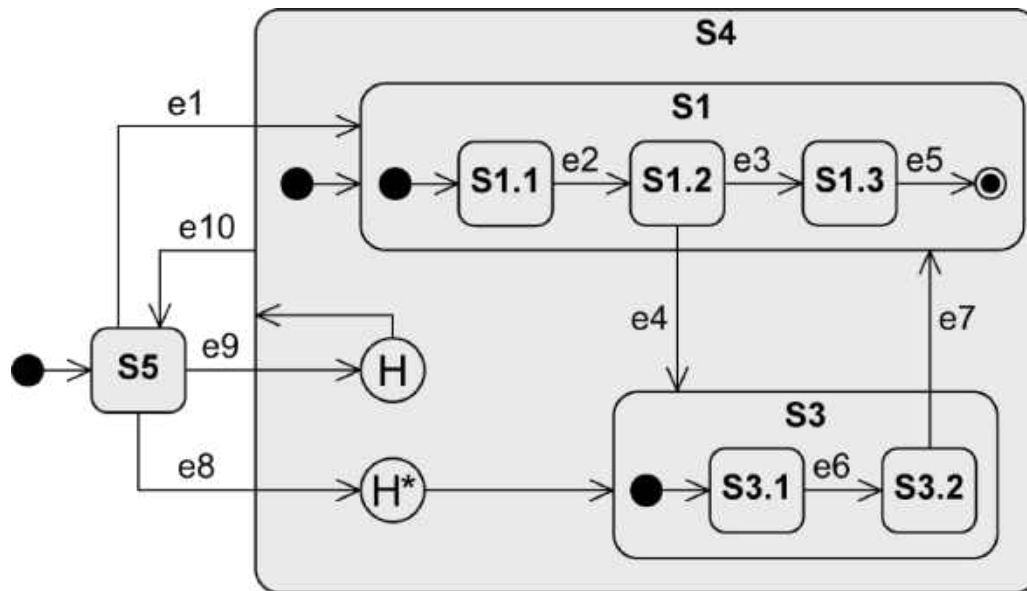
- Mengingat substate mana dari sebuah composite state yang terakhir kali aktif
- Mengaktifkan substate “lama” dan semua entry activity dijalankan secara sekuensial dari luar ke dalam composite state
- Tepat satu edge keluar dari history state menuju ke sebuah substate yang digunakan jika
  - composite state belum pernah aktif sebelumnya
  - composite state keluar melalui state akhir
- Shallow history state menyimpan state pada level yang sama dalam composite state



- Deep history state menyimpan substate yang terakhir aktif dari keseluruhan substate

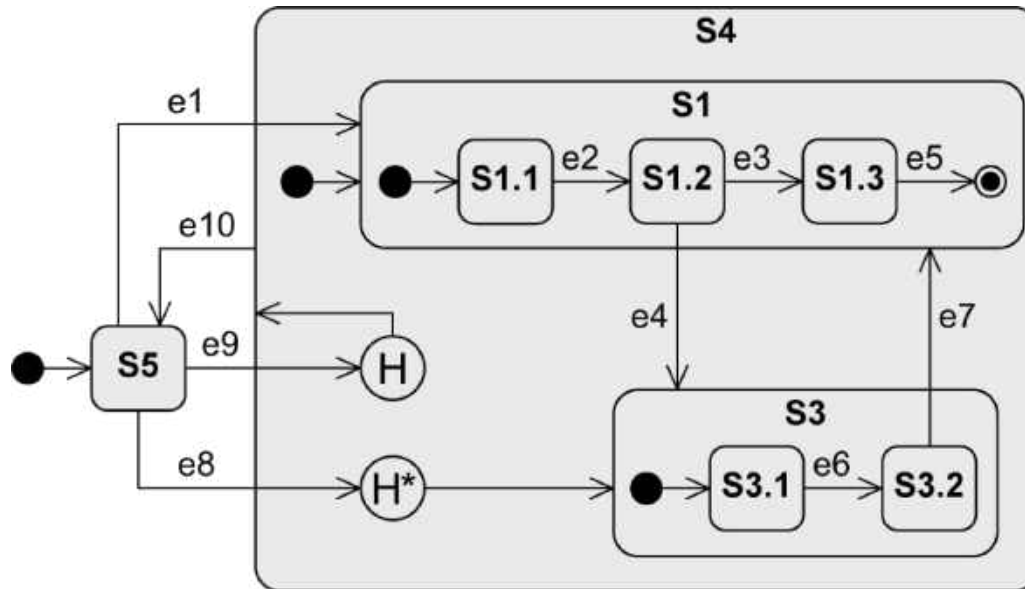


## Contoh: History State (1/4)



Event	State
„Beginning“	S5
e1	S4/S1/S1.1
e2	S1.2
e10	S5
e9	(H→) S1/S1.1

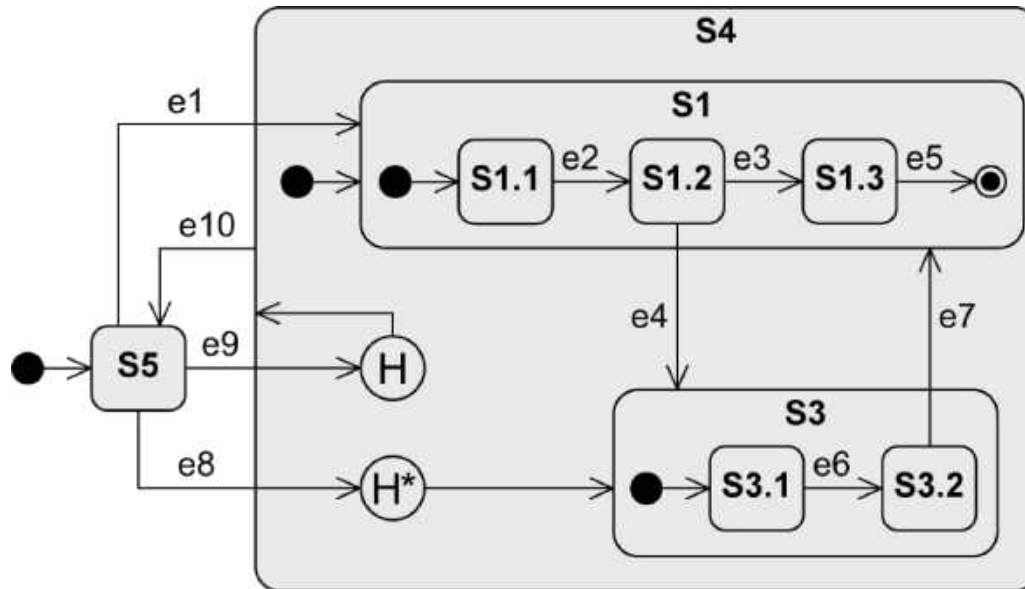
## Contoh: History State (2/4)



Event	State
„Beginning“	S5
e1	S4/S1/S1.1
e2	S1.2
e10	S5
e8	(H*→) S1.2

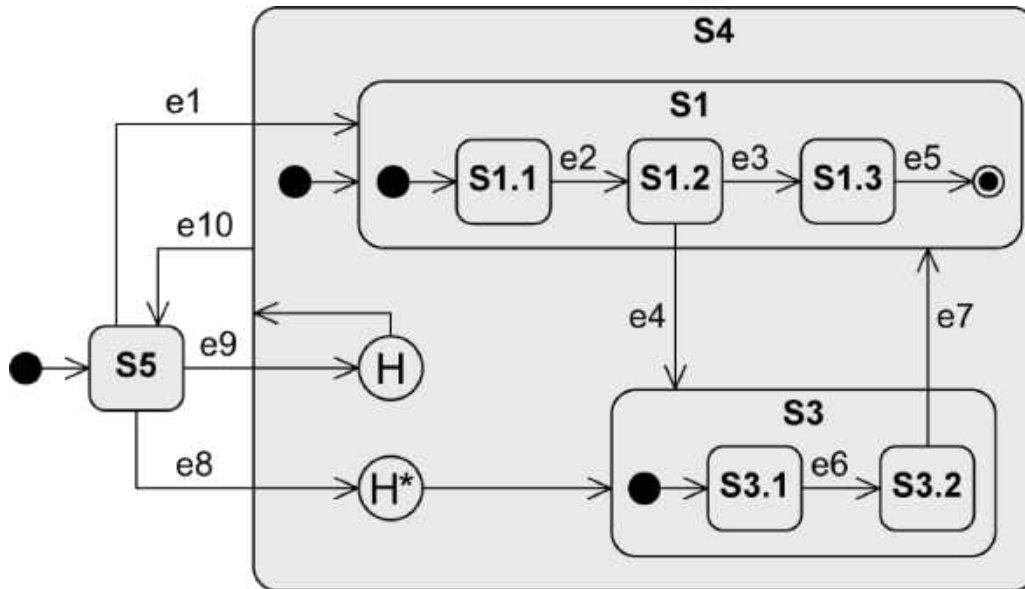


## Contoh: History State (3/4)



Event	State
„Beginning“	S5
e9	(H→) S1/S1.1

## Contoh: History State (4/4)

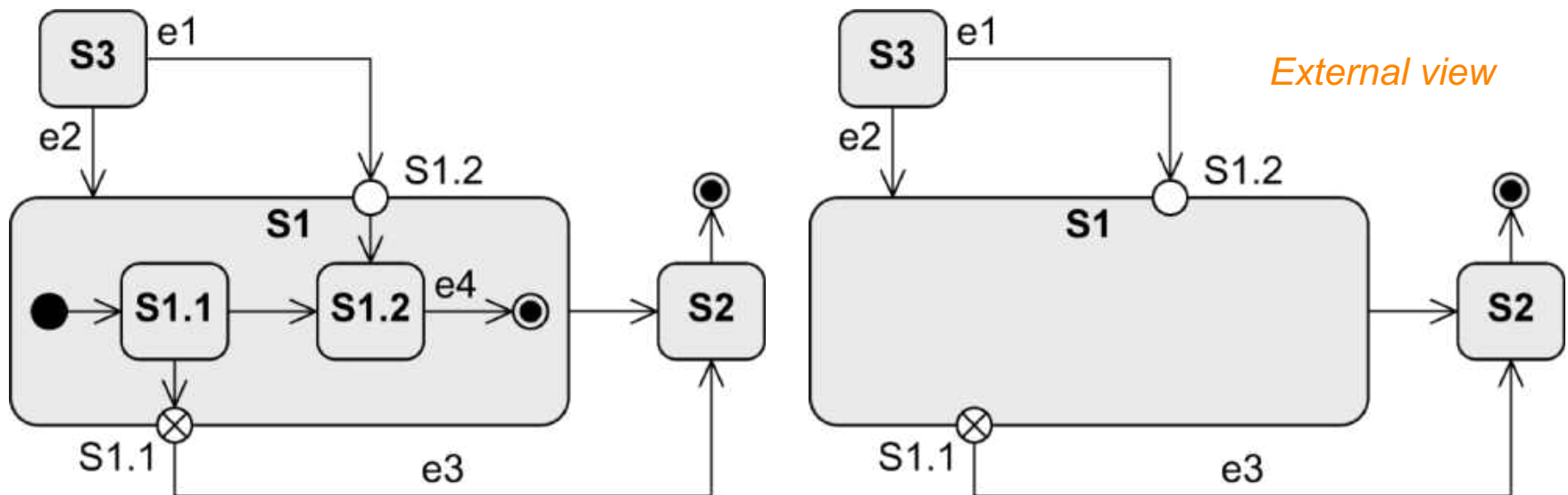


Event	State
„Beginning“	S5
e8	(H* →) S3/S3.1

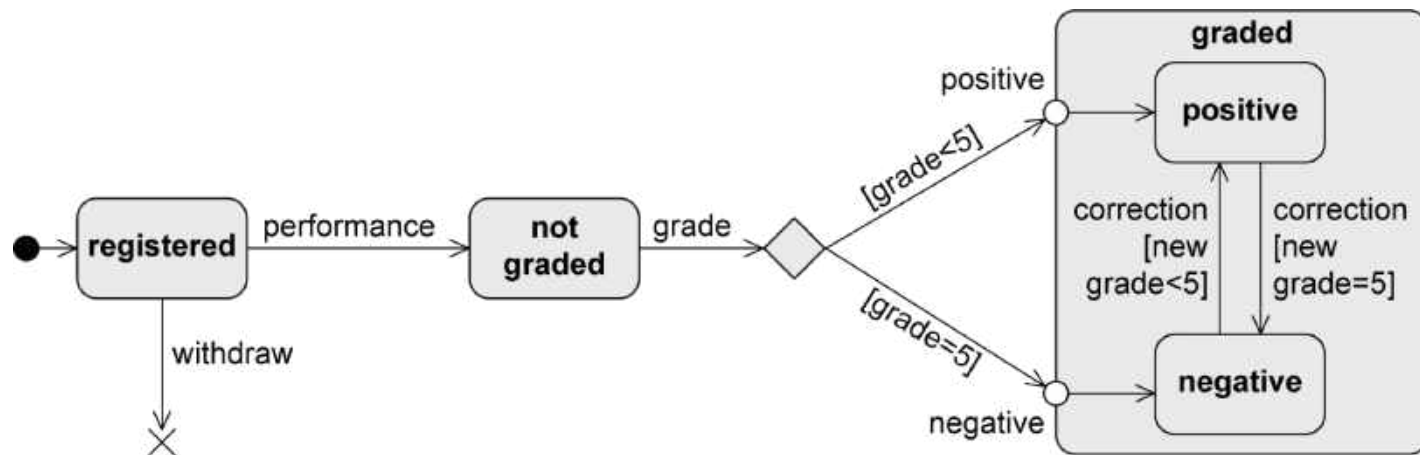
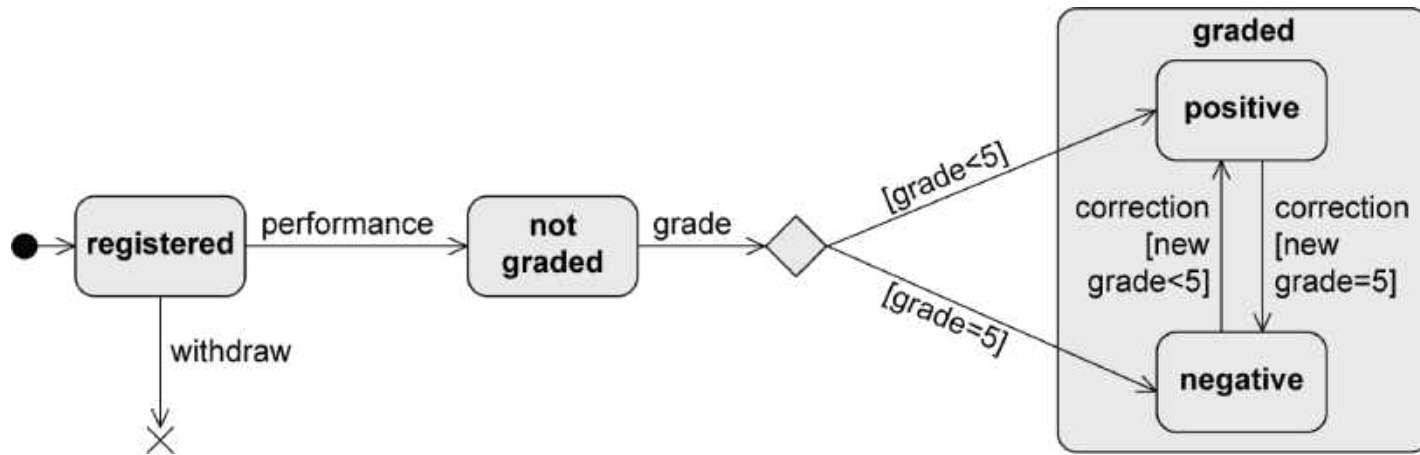
# Entry dan Exit Points

## ■ Mekanisme encapsulation/enkapsulasi

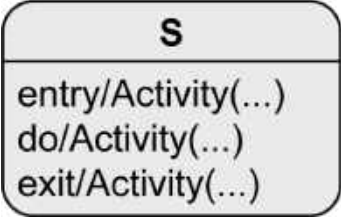
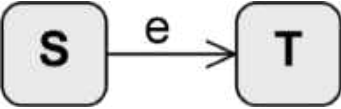



- Sebuah composite state dapat dimasuki atau keluar melalui sebuah state selain state awal dan state akhir
- External transition tidak harus/tidak perlu mengetahui struktur composite state



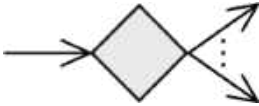
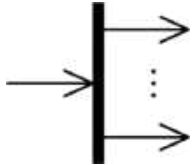
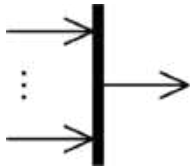
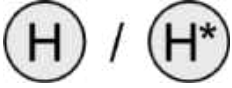
## Contoh: Entry dan Exit Points



## Notasi Elemen (1/2)

Name	Notation	Description
State		Deskripsi dari sebuah “rentang waktu” spesifik dimana sebuah object berada dalam “masa hidup”-nya. Dalam sebuah state, aktivitas dapat dijalankan oleh object.
Transition		State transition e dari state awal S menuju ke state akhir T
Initial state		Awal dari state machine diagram
Final state		Akhir dari state machine diagram
Terminate node		Terminasi (berakhirnya) state machine diagram sebuah object

## Notasi Elemen (2/2)

Name	Syntax	Beschreibung
Decision node		Node dari beberapa alternative transition dapat berasal
Parallelization node		Membagi sebuah transition menjadi beberapa transition yang paralel
Synchronization node		Menggabungkan beberapa transition yang paralel menjadi sebuah transition
Shallow / deep history state		“Alamat pengembalian” ke sebuah substate atau sebuah nested substate dalam sebuah composite state